Universitat de Lleida

TREBALL FINAL DE GRAU

ESCOLA
POLITÈCNICA SUPERIOR
UNIVERSITAT DE LLEIDA
INSPIRING THE FUTURE

Estudiant: **Joaquim Picó Mora**

Titulació: Grau en Enginyeria Informàtica

Títol de Treball Final de Grau: Implementació d'una eina per l'obtenció i l'anàlisi de discussions en la xarxa social YouTube

Director/a: Ramon Bejar Torres

Presentació
Mes: Juliol

Any: 2021

Preàmbul

El següent treball s'ha realitzat amb l'objectiu d'implementar una eina que permeti i faciliti l'obtenció de l'arbre de discussió que es genera en els comentaris d'un vídeo de YoutTube.

Després d'un primer anàlisi sobre la informació que ens ofereix la seva API, s'ha detectat que YouTube aplica una simplificació a l'hora d'emmagatzemar els comentaris dels usuaris fent que es perdi informació rellevant per tal de crear l'arbre de discussió. Quan YouTube guarda els comentaris pertanyents a un fil, assigna com a pare de tots aquests el principal. Fet que, si hi ha algun comentari que contesta a un altre se li assigni com a pare el principal en canvis del que contesta. Així doncs, s'ha pensat a implementar una solució que pugui estar a l'abast de tothom en cas que algú es trobi amb el mateix problema, i és per això que s'ha decidit desenvolupar una llibreria. Aquesta s'ha programat amb el Llenguatge de Programació Python i posteriorment s'ha publicat a PyPi org.

Per posar solució a aquest conflicte de falta d'informació a l'hora de crear l'arbre de discussió, s'ha dissenyat un algoritme automàtic mitjançant el qual es realitzarà un rànquing dels candidats a ser el pare d'un comentari i s'escollirà aquell més rellevant.

Agraïments

Aquest treball no hauria estat possible sense el meu tutor Ramón Bejar Torres, amb el qual sota la seva supervisió vam escollir aquest tema i vaig començar amb aquest projecte.

De la mateixa manera, m'agradaria agrair al grup de recerca GREiA per deixar-me participar en el desenvolupament d'un mòdul per al seu projecte d'anàlisi de discussions en xarxes socials.

Finalment, m'agradaria agrair a la meva família i parella els quals m'han donat suport al llarg de la realització d'aquest treball, així com també als meus amics que a part de suport, m'han donat consells molt valuosos i la seva opinió sincera en tot moment.

Índex

Índex de Figures **iv**

Índex de Taules **v**

1 Introducció **vi**

1.1 Context vii

1.2 Motivació viii

1.3 Objectius ix

2 Anàlisi **x**

2.1 Requeriments x

2.2 Estructura d'arbre x

2.3 Recollida de dades xi

2.3.1 Youtube Data API xi

2.3.2 Scraper xvii

2.4 Canvi de requeriments xvii

3 Disseny i Implementació **xix**

3.1 Decisions de disseny xix

3.1.1 Llenguatge xxi

3.1.2 GitHub xxi

3.1.3 Tests Automàtics i Coverage xxii

3.1.4 Paquet Pypi xxii

3.2 Decisions d'implementació xxii

3.2.1 YoutubeDiscussionTreeAPI xxii

3.2.2 YoutubeDiscussionTree xxiv

3.2.3 Interaccions amb l'API i Gestió de la Quota xxvii

3.2.4 Algoritme de Resolució de Conflictes xxix

4 Ús del sistema **xxxiii**

4.1 Instal · lació xxxiii

4.2 API xxxiii

4.2.1 Generació de l'arbre xxxiv

4.2.2 Serialització de l'arbre xxxiv

4.2.3 Representació de l'arbre xxxv

4.2.4 Llista de Nodes xxxv

4.2.5 Búsqueda de vídeos xxxv

4.2.6 Consultar Quota xxxvi

5 Avaluació **xxxvii**

6 Conclusions **xli**

7 Bibliografia **xliii**

Índex de Figures

1 Disseny Sistema xx

2 YoutubeDiscussionTreeAPI xxii

3 YoutubeDiscussionTree xxiv

4 QuotaManager xxvii

5 _http module xxviii

6 Cosine Similarity xxxiii

7 Comentari Principal del Fil xxxvii

8 Resposta Normal xxxviii

9 Resposta amb etiquetatge xxxviii

10 Resposta amb etiquetatge i conflicte xxxix

Índex de Taules

1 Youtube Data API Operations xii
2 Youtube Data API CommentThread Parameters xii
3 Youtube Data API Video parameters xv
4 YouTube Data API Search parameters xix

1 Introducció

Juntament amb l'auge de la web 2.0, es va originar un nou concepte tecnològic anomenat xarxes socials. Aquest es basa en plataformes interactives que permeten la creació i intercanvi d'informació, idees, interessos i moltes altres formes d'expressió via comunitats virtuals i xarxes d'amistat.

La primera xarxa social es data del 1972, de la mà de PLATO system, els quals van desenvolupar un estil de xarxa social amb la terminal com a interfície que contenia fòrums, xats, videojocs... No ha sigut, però, fins al segle 21 que les xarxes socials han agafat un paper més rellevant dins la societat. Amb la sortida de plataformes com Yahoo, Twitter i la més important i que va fer enlairar el sector, Facebook. A més, amb el naixement a principis de segle de l'indústria de l'smartphone, el nombre d'usuaris de les xarxes socials ha crescut a nivells incommensurables, i n'han nascut moltes de noves durant el camí, com ara: Instagram, Twitter, Reddit, Linkedin...

A nivell conceptual, les xarxes socials són originades en termes de la comunicació. Es refereix al conjunt de grups, comunitats i organitzacions vinculats els uns als altres a través de relacions socials. Com ja s'ha comentat, avui dia a internet existeixen moltes d'aquestes plataformes que implementen aquest tipus de comunicació. Aquesta succeeix quan els usuaris publiquen contingut per a altres usuaris de la seva xarxa, de forma que els segons hi poden interaccionar. Donat que, aquestes xarxes socials tenen un nombre exorbitant d'usuaris i milions d'interaccions cada dia, s'han convertit en una font molt rica de dades i en espais molt interessants d'investigar.

El grup de recerca de GREIA de la Universitat de Lleida ha estat treballant recentment en una eina d'anàlisis de discussions que es produeixen en xarxes socials, la qual han aplicat a xarxes com Reddit i Twitter. A partir d'un

arbre de discussió, aquesta eina és capaç de determinar qui va guanyant en un debat o quina és l'opinió més forta.

En aquest projecte s'ha treballat en el desenvolupament d'una eina per tal d'integrar una altra xarxa a aquest analitzador de discussions. Així doncs, en aquest document es recull tota la informació referent al desenvolupament d'aquesta tasca.

S'ha dividit el document en sis parts, primerament una introducció al projecte on es defineix el context, les motivacions i l'objectiu d'aquest treball. Seguidament es realitzara una descripció de l'anàlisi que s'ha dut a terme per desenvolupar-lo. Tot seguit s'explicarà el disseny i la implementació i es documentarà l'ús del sistema implementat. I Finalment, es comentaran els resultats obtinguts després d'una avaluació de la solució i és conclourà el projecte amb un apartat de conclusions finals on es realitzarà una reflexió sobre tot el procés.

1.1 Context

YouTube és una xarxa social d'origen estatunidenc dedicada a què els seus usuaris hi compateixin vídeos. Va ser creada per 3 antics empleats de Paypal el febrer de 2005 i l'octubre de 2006 va ser adquirida per Google per 1650 milions de dòlars. Actualment, és el lloc web més utilitzat de la seva categoria.

La idea de YouTube va sorgir a causa de les dificultats que van experimentar els tres creadors a l'hora d'intentar compartir entre ells els vídeos d'una festa a San Francisco. El domini va ser activat el 14 de Febrer de 2005 i el 23 d'abril del mateix any es va publicar el primer vídeo, "Me at the Zoo".

La comunicació a YouTube succeeix quan un usuari interacciona sigui reaccionant o comentant sobre un vídeo que ha publicat un altre usuari. Cada vídeo publicat a la plataforma, disposa d'una secció dedicada a què els usuaris puguin comentar i discutir entre ells sobre el contingut del vídeo. Per altra banda, permet reaccionar tant al vídeo com als comentaris mitjançant likes i dislikes.

Quant a números, YouTube compta amb la increïble xifra de més de dos bilions d'usuaris repartits en 100 països que poden gaudir de contingut en 80 llengües, tot aglomerant diàriament un total de més d'un bilió d'hores de visualització de contingut de la plataforma.

Al tractar-se d'una xarxa tan gran, implica que diàriament molts usuaris hi publiquin vídeos i molts d'altres els vegin i hi reaccionin, fet que fa de YouTube un entorn d'estudi molt interessant amb un nombre gegant de dades.

1.2 Motivació

Tot i que YouTube no és una xarxa orientada 100% a la discussió com ho poden ser Twitter i Reddit, en els comentaris dels vídeos s'hi poden originar discussions de temes molt interessants relacionats i/o tractats en aquests. A més, com bé ja s'ha dit, actualment és una de les xarxes més utilitzades i amb més usuaris i hores de visualització de contingut del món.

S'ha realitzat un anàlisi per examinar de quina forma aconseguir les dades de YouTube, escollir quines dades processar, processar-les i posteriorment construir un arbre de discussió que relaciona els comentaris segons a qui responen. Tot això utilitzant tècniques de processament i transformació de dades.

A més, no només es tracta de motivació tecnològica, sinó que també hi ha un incentiu social, ja que, es vol implementar una solució general per tal de facilitar la tasca a tothom que es trobi amb el mateix requeriment en el seu projecte.

Addicionalment, s'afegeix la motivació personal pel fet d'ajudar a un projecte del grup de recerca de la Universitat de Lleida com és GREiA.

1.3 Objectius

L'objectiu d'aquest treball és implementar una solució que compleixi els requeriments de l'analitzador de discussions per a treballar sobre les discussions que es generen en els comentaris dels vídeos publicats a YouTube, així com realitzar una implementació suficientment general perquè pugui ser aprofitada per altres persones que s'enfrontin al mateix requeriment en els seus projectes.

Aquesta eina haurà de ser capaç de facilitar al màxim la tasca a l'hora de construir un arbre de discussió totalment informat a partir de les dades que es puguin extreure de YouTube.

2 Anàlisi

2.1 Requeriments

Abans de començar amb el disseny de la solució, es realitza un anàlisi dels requeriments sobre els quals es vol implementar. S'ha de dissenyar i implementar una eina que compleixi:

- Script usable mitjançant paràmetres d'entrada
- Aconseguir els comentaris i informació d'un vídeo de YouTube
- Serialitzar la discussió amb el format que entén l'analitzador

2.2 Estructura d'arbre

Tota estructura d'arbre esta composta per un node arrel, del qual en surten branques que desencadenen en nodes fulla. Amb xarxes com Reddit o Twitter el node pare pot ser el primer post, i, les branques i les fulles les discussions que es generen a partir dels comentaris d'aquell post. Amb YouTube és una mica més complex, ja que, el primer post no és text sinó un vídeo.

S'han valorat diferents opcions, la primera i aparentment més senzilla, és agafar el contingut de la descripció del vídeo com a primer post. El problema d'aquesta opció resideix en què en la majoria de vídeos de la plataforma, a la descripció del vídeo, no s'hi troba una descripció d'aquest. Per tant aquesta opció no és vàlida a l'hora d'obtindre context del que es parla al vídeo.

Com a alternativa, s'ha pensat a utilitzar una transcripció de tot el contingut d'aquest. És a dir, mitjançant una eina, en el nostre cas una llibreria publica anomenada YouTube Transcript API [4], transcriure tot el que es diu en el vídeo. Aquesta solució ens dóna tot el context que necessitem del

vídeo, ja que transcriu paraula per paraula tot el que s'hi diu. L'eina en qüestió, recull els subtítols del vídeo, ja siguin autogenerats o creats per l'usuari. Cal remarcar que YouTube en cas que no hi hagi subtítols creats per l'usuari, detecta l'idioma que es parla en el vídeo i n'autogenera uns. Per tant, tots els vídeos en què el contingut sigui en anglès contindran una transcripció en anglès.

Tot i així aquesta solució presenta alguns problemes. Un d'ells és que no som capaços d'extreure context d'un vídeo en el qual no hi hagi contingut parlat, ja que no hi existirà cap transcripció. Per altra banda, si els vídeos són molt llargs i hi ha molt contingut, es poden generar transcripcions molt grans. Més endavant s'explicarà com es tracten aquests problemes.

Així doncs, l'arbre estarà compost per un primer node arrel que contindrà una transcripció del contingut de vídeo, i penjant d'aquest tots els fils de comentaris que hi té associats.

2.3 Recollida de dades

Quant a la recollida de dades, també s'han contemplat diverses opcions. Primerament es va contemplar l'opció d'utilitzar la Youtube Data API, i seguidament l'opció d'implementar un scraper.

2.3.1 Youtube Data API

La primera d'elles és aconseguir la informació a través de l'API de YouTube. La YouTube Data API [10] és un servei web que oferix YouTube als seus usuaris per interaccionar amb la plataforma de forma programàtica. Aquesta ofereix diferents grups d'operacions que actuen sobre recursos determinats de la plataforma. En el cas que ocupa aquest treball, com es pot verue en la taula 1, interessen les operacions relacionades en obtindre informació sobre

els vídeos i els comentaris d'aquests.

Nom	Operacions	Descripció
CommentThreads	list, insert, update	Informació sobre fils de comentaris.
Search	list	Cerca de recursos a través de paràmetres de l'API
Videos	list, insert, update, rate, getRating, delete	Informació sobre vídeos de YouTube

Taula 1: Youtube Data API Operations

CommentThreads

Per tal d'aconseguir els fils de comentaris dels vídeos, es farà una petició GET a l'endpoint de llistat del grup CommentThreads [11]. Juntament amb la petició, s'hi passen un seguit de paràmetres per tal d'indicar quina informació es vol rebre d'aquests així com per filtrar els resultats. Els paràmetres que s'utilitzen, són els definits en la taula 2.

Nom	Descripció	Values
Obligatoris		
part	Propietats que es volen llistar dels commentThreads	id, replies, snippet
Filtres		
videoId	Retorna la llista de commentThreads associats a un vídeo	string
maxResults	Nombre de commentThreads que es desitgen	unsigned integer desde 1 fins a 100, per defecte 20
order	Ordre en què es llisten els resultats	time, relevance

Taula 2: Youtube Data API CommentThread Parameters

D'aquesta petició, es rep com a resposta un objecte json:

```
{
```

```
  "kind": "youtube#commentThreadListResponse",
  "etag": etag,
  "nextPageToken": string,
  "pageInfo": {
    "totalResults": integer,
    "resultsPerPage": integer
  },
  "items": [
    commentThread Resource
  ]
}
```

Es pot veure que al camp ítems s'obté una llista de commentThread Resources els quals seran també objectes json:

```
    {
  "kind": "youtube#commentThread",
  "etag": etag,
  "id": string,
  "snippet": {
    "channelId": string,
    "videoId": string,
    "topLevelComment": comments Resource,
    "canReply": boolean,
    "totalReplyCount": unsigned integer,
    "isPublic": boolean
  },
  "replies": {
    "comments": [
      comments Resource
    ]
  }
}
```

On tant el topLevelComment com la llista replies estan compostos per comments Resources:

```
{
  "kind": "youtube#comment",
  "etag": etag,
  "id": string,
  "snippet": {
    "authorDisplayName": string,
```

```
    "authorProfileImageUrl": string,
    "authorChannelUrl": string,
    "authorChannelId": {
      "value": string
    },
    "channelId": string,
    "videoId": string,
    "textDisplay": string,
    "textOriginal": string,
    "parentId": string,
    "canRate": boolean,
    "viewerRating": string,
    "likeCount": unsigned integer,
    "moderationStatus": string,
    "publishedAt": datetime,
    "updatedAt": datetime
  }
}
```

D'aquesta forma, s'obtindrà una llista de commentThreads, els quals estan compostos d'un topLevelComment, que representa el comentari principal del fil de comentaris i una llista de respostes associades a aquest. En aquest pas s'ha detectat que YouTube aplica una simplificació de les dades a l'hora d'emmagatzemar-les. Des de la UI de la plataforma, es pot contestar a una resposta que pertany a un topLevelComment, però YouTube al guardar el comentari assignara com a parentId la id del topLevelComment en canvis de la resposta a la que estem contestant. Per altra banda, fica al camp de text un tag "@nom de l'usuari". En aquest pas, es perd informació rellevant a l'hora de construir l'arbre de discussió, ja que no hi ha forma de saber a quin comentari està contestant realment si l'usuari a qui es refereix el tag "@nom de l'usuari" ha realitzat més d'una participació en la discussió prèviament.

Vídeos

Per tal d'aconseguir informació referent al vídeo s'utilitzarà l'operació de llistat del grup Vídeos [13]. De la mateixa manera que el llistat en el cas dels comments threads, es realitzarà una petició GET a un enpoint passant-hi

uns paràmetres per escollir la informació que es vol rebre i filtrar el resultat.

Nom	**Descripció**	**Values**
Obligatoris		
part	Propietats que defineixen els valors que es volen aconseguir del vídeo	id, snippet, contentDetails, fileDetails, player, processingDetails, recordingDetails, statistics, status, suggestions, topicDetails
Filtres		
id	Id del vídeo del qual es vol aconseguir informació	string

Taula 3: Youtube Data API Video parameters

Es rep com a resposta el mateix que el que es rep al realitzar una petició del recurs commentThreads, però, aquest cop, en canvis d'una llista de commentThreads es rep una llista de recursos Vídeo. En aquest cas, a l'especificar una única id es rep una llista amb només un element.

```
{
  "kind": "youtube#videoListResponse",
  "etag": etag,
  "nextPageToken": string,
  "prevPageToken": string,
  "pageInfo": {
    "totalResults": integer,
    "resultsPerPage": integer
  },
  "items": [
    video Resource
  ]
}
```

On el recurs vídeo es representa com:

```
{
  "kind": "youtube#video",
  "etag": etag,
  "id": string,
```

```
  "snippet": {
    "publishedAt": datetime,
    "channelId": string,
    "title": string,
    "description": string,
    "thumbnails": {
      (key): {
        "url": string,
        "width": unsigned integer,
        "height": unsigned integer
      }
    },
    "channelTitle": string,
    "tags": [
      string
    ],
    "categoryId": string
  },
  "statistics": {
    "viewCount": unsigned long,
    "likeCount": unsigned long,
    "dislikeCount": unsigned long,
    "favoriteCount": unsigned long,
    "commentCount": unsigned long
  }
 }
```

D'aquesta forma s'obté tota la informació necessària per crear el node arrel. Tot i poder aconseguir la informació de YouTube de forma relativament senzilla, aquesta solució presenta alguns defectes. Primerament, la YouTube Data API limita el seu ús de forma gratuïta a una quota diària de 10000 unitats. La quota és la unitat que es fa servir per determinar l'ús de l'api, cada operació consumeix un nombre de quota. En el cas dels llistats, que és l'operació que s'utilitza tant quan volem aconseguir la informació dels fils de comentaris com dels vídeos, el cost és 1. Per altra banda, s'ha detectat que YouTube simplifica les dades dels comentaris a l'hora de guardar-les, assignant com a parentId de totes les respostes que formen part d'un commentThread l'id del topLevelComment d'aquest. Aquestes limitacions han conduït a l'exploració d'altres possibles tècniques per aconseguir la infor-

mació que necessitem de YouTube.

2.3.2 Scraper

Es va començar a explorar la possibilitat d'utilitzar un scraper que agafés la informació dels comentaris. Aquesta solució es va acabar descartant, ja que oferia més contres que l'opció de l'API. Per una banda, l'scraper era molt menys escalable perquè depèn que la pàgina no canviï, en cas de canviar aquest deixaria de funcionar. Per altra banda, el cost de realitzar un scraper és molt més elevat que realitzar peticions a l'API, s'hauria de dedicar més esforç per intentar aconseguir el llistat de commentThreads. A més, tampoc es pot aconseguir la mateixa quantitat d'informació que pots aconseguir via API. L'únic benefici que ofereix és que en cas d'aconseguir un scraper que donés la informació suficient per a generar un arbre, aquest oferiria la possibilitat de generar-ne sense limitacions. De totes maneres, amb aquesta implementacio s'hi troba el mateix conflicte que s'ha esmentat anteriorment de la simplificació de les dades a l'hora de guardar la referència al comentari pare de les respostes. A més, es considera que el cost de les operacions de llistat és molt baix i per tant es poden arribar a generar molts arbres de discussió diàriament.

2.4 Canvi de requeriments

Com que s'ha detectat aquesta simplificació de les dades a l'API de YouTube i és un problema que pot afectar a més persones que intentin treballar sobre les discussions que s'hi generen, s'ha decidit realitzar un canvi de requeriments de forma que s'implementarà una solució més general. Aquesta, ha de poder ser utilitzada de forma senzilla per qualsevol persona que tingui aquest requeriment en el seu projecte. Per tal de resoldre aquests conflictes s'implementara un algoritme de resolució automàtic, tal que, a partir del tag "@nom de l'usuari" agafara tots els possibles candidats a ser el pare de

la resposta i donarà com a resultat el pare corresponent.

Els nous requeriments ara seran:

- Es deixa de banda la idea de programar un script, ja que s'ha de dissenyar una solució més general
- Fàcil d'utilitzar
- Fàcil d'accedir
- Aconseguir els comentaris i informació d'un vídeo de YouTube
- Construcció d'una estructura de dades per representar la discussió
- Serialitzar l'arbre de discussió en un format general
- Control de la quota consumida per l'api
- Facilitat d'ús en processos automàtics de generació d'arbres
- Algoritme de resolució de conflictes automàtic

Com es pot veure, la llista de requeriments ha augmentat. El que es busca és una solució general, la qual ha de ser fàcil d'utilitzar i d'accedir. A més, s'ha detectat que Google no dóna la informació sobre la quantitat de quota de l'API que s'ha consumit, així que s'haurà de pensar en una solució per tal de controlar quanta quota gasta l'usuari en utilitzar la llibreria. A més, es vol facilitar l'ús d'aquesta en processos de generació d'arbres automàtics.

Per tal de complir aquest últim requeriment s'afegira a la solució la possibilitat de realitzar cerques de vídeos per tal d'obtindre la seva informació i posteriorment poder generar grups d'arbres. Mitjançant l'operació de cerca [12] que ofereix la YouTube Data API, es poden cercar recursos de forma acotada tal com es realitzen amb el buscador de la plataforma. Podem

veure a la taula 4 els paràmetres que s'utilitzaran per a realitzar la petició a l'endpoint de Search de l'API.

Nom	**Descripció**	**Values**
Obligatoris		
part	Propietats que defineixen els valors que es volen aconseguir de la cerca	id, snippet
Opcionals		
maxResults	Determina el nombre màxim de recursos de la cerca	0-50, default 5
q	Termes de cerca a consultar	string
type	Especifica el tipus de recurs que es vol en la llista resultant	channel, playlist, video
order	Determina l'ordre en què es retornen els resultats	date, rating, relevance, title, videoCount, viewCount
videoCapiton	Filtra el resultat segons si te subtítols o no	any, closedCaption, none

Taula 4: YouTube Data API Search parameters

Tot i que el cost d'aquesta operació és una mica elevat (100 unitats), es pot aconseguir informació de fins a 50 vídeos de cop. Cosa que permet realitzar algoritmes d'automatització de generació d'arbres en cas de voler aconseguir un volum de dades més elevat.

3 Disseny i Implementació

3.1 Decisions de disseny

S'ha decidit que per tal d'oferir una solució general que sigui usable i accessible, es dissenyarà una llibreria que encapsuli la implementació així com

la gestió de les dependències d'aquesta. Les principals funcionalitats que haurà de contenir són:

- Generació d'un arbre de discussió
- Cerca de vídeos
- Control de la quota consumida
- Serialització de l'arbre
- Visualització de l'arbre

Per tal de poder utilitzar les funcionalitats que ofereix la implementació de forma còmoda, s'ha pensat amb un disseny per al sistema el qual utilitza un objecte com interfície per interactuar amb les funcionalitats de la llibreria. Aquest disseny està representat en la figura 1.

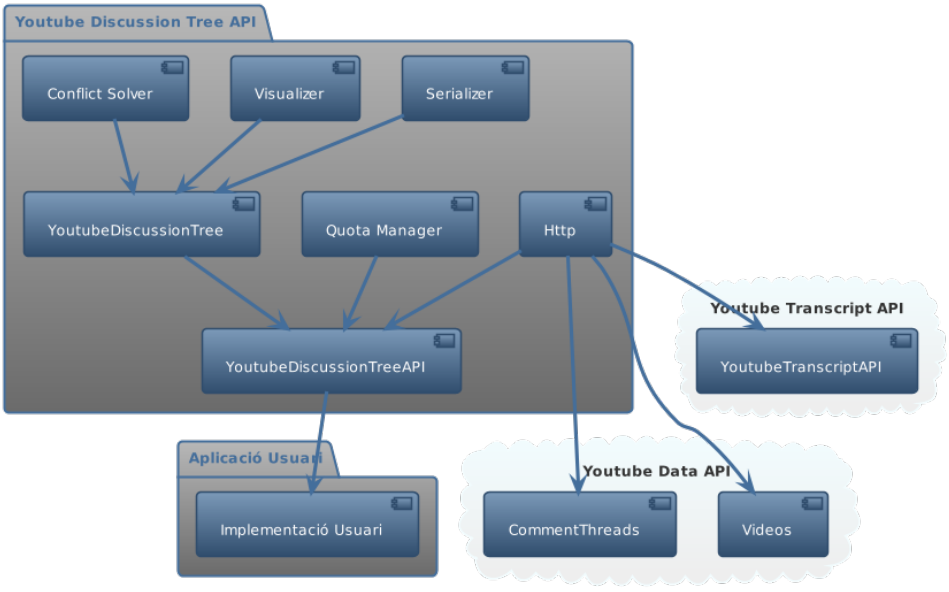

Figura 1: Disseny Sistema

Es pot observar que l'objecte que serveix per interactuar amb les funcionalitats de tota la llibreria és YoutubeDiscussionTreeAPI. Per altra banda,

es compta amb l'objecte YoutubeDiscussionTree per tal de crear i emmagatzemar l'estructura d'arbre que representa la discussió. Un altre element a destacar és el mòdul http, el qual serà l'encarregat de realitzar totes les comunicacions amb els serveis externs a l'API mitjançant protocol HTTP. Addicionalment, l'objecte YoutubeDiscussionTree comptarà amb mòduls per tal de realitzar la seva serialització i representació, així com un que contindrà l'algoritme de resolució automàtic dels conflictes que presenta l'API de YouTube, els quals hem exposat prèviament. Finalment, QuotaManager serà l'objecte encarregat d'actualitzar i mantenir la quota que es gasta mitjançant la llibreria.

3.1.1 Llenguatge

Quant al llenguatge escollit per al desenvolupament de la llibreria, Python és el candidat perfecte. Actualment és un dels llenguatges de programació que està creixent de forma més ràpida, tant en comunitat com en desenvolupament d'eines i paquets útils. A més, en el món de la intel · ligència artificial i de l'anàlisi i transformació de dades és un llenguatge que es fa servir molt, així que el fet d'implementar-ho en Python pot fer que arribi a més gent.

3.1.2 GitHub

Tot el desenvolupament es realitza mitjançant el software de control de versions GitHub, la llibreria residirà en un repositori públic des del qual els usuaris es podran descarregar el codi font. Aquesta eina també permet que els usuaris interactuïn directament amb el mantenidor del projecte, de forma que si troben algun error de programació o es creu que es pot implementar algun mòdul més a la llibreria ho poden comunicar publicant una "issue".

3.1.3 Tests Automàtics i Coverage

Per tal de mantenir la validesa del software a mesura que es van afegint codi, s'ha configurat al repositori un sistema automàtic de validació. Mitjançant l'eina Travis [2] combinada amb l'eina Coveralls [3], quan es realitzi un push a la branca principal del projecte a github, es llençarà un procés que executarà els testos per validar que no s'ha pujat codi defectuós. A la vegada es comprovarà el percentatge de línies de codi que s'ha provat en aquest procés.

3.1.4 Paquet Pypi

L'Índex de Paquets de Python o també dit Pypi, és el repositori de software oficial per aplicacions de tercers en el llenguatge de programació Python. Un cop realitzada la primera release de la llibreria, aquesta es penjarà a aquest repositori per tal de facilitar la seva descàrrega i ús als usuaris que la vulguin utilitzar.

3.2 Decisions d'implementació

3.2.1 YoutubeDiscussionTreeAPI

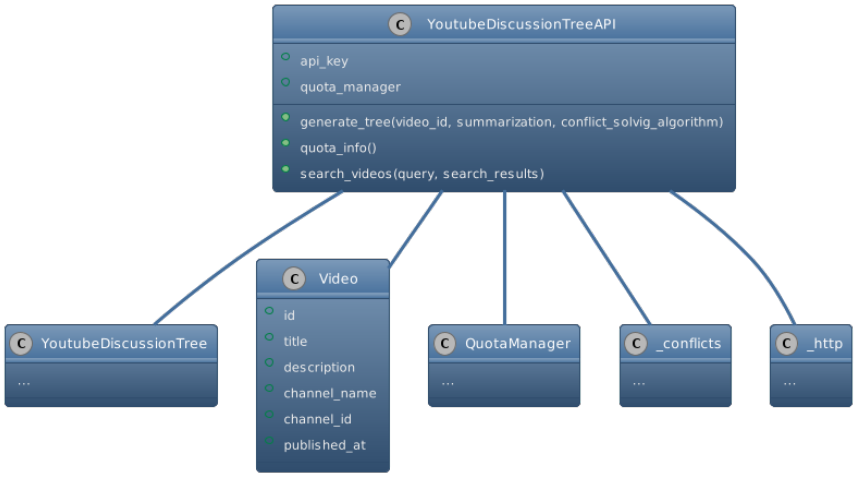

Figura 2: YoutubeDiscussionTreeAPI

Per tal de facilitar-ne l'ús, s'ha pensat en un objecte de presentació des del qual es farà crida a les principals funcionalitats de la llibreria. Aquest, es construirà a partir del valor de la clau d'API del projecte del consumidor, que posteriorment serà la que s'utilitzarà per interaccionar amb la YouTube Data API. Com es pot veure al diagrama, aquest objecte està compost per tres mètodes, el de generació de l'arbre, el de cerca de vídeos, i un altre per consultar el número de quota diària que s'ha consumit.

Quant al mètode de generació de l'arbre, s'ha pensat com una funció d'alt nivell que rep per paràmetre l'algoritme de resolució dels conflictes que s'esmentava anteriorment. D'aquesta forma, es pot rebre per paràmetre una funció que, donat un llistat de candidats i una resposta, retorna l'id del candidat al qual es creu que respon. Si no s'assigna cap valor al paràmetre "conflict_solving_algorithm", s'utilitzarà per defecte l'algoritme de resolució de conflictes automàtic implementat a la llibreria.

Per altra banda per solucionar el problema en què, al generar el node arrel, hi podia haver transcripcions del contingut dels vídeos de longitud molt gran, s'ha pensat a afegir un altre paràmetre booleà per decidir si es vol realitzar un resum de la transcripció. Aquest resum es realitza mitjançant un model entrenat que proporciona la llibreria Transformers de The HuggingFace Project [7].

3.2.2 YoutubeDiscussionTree

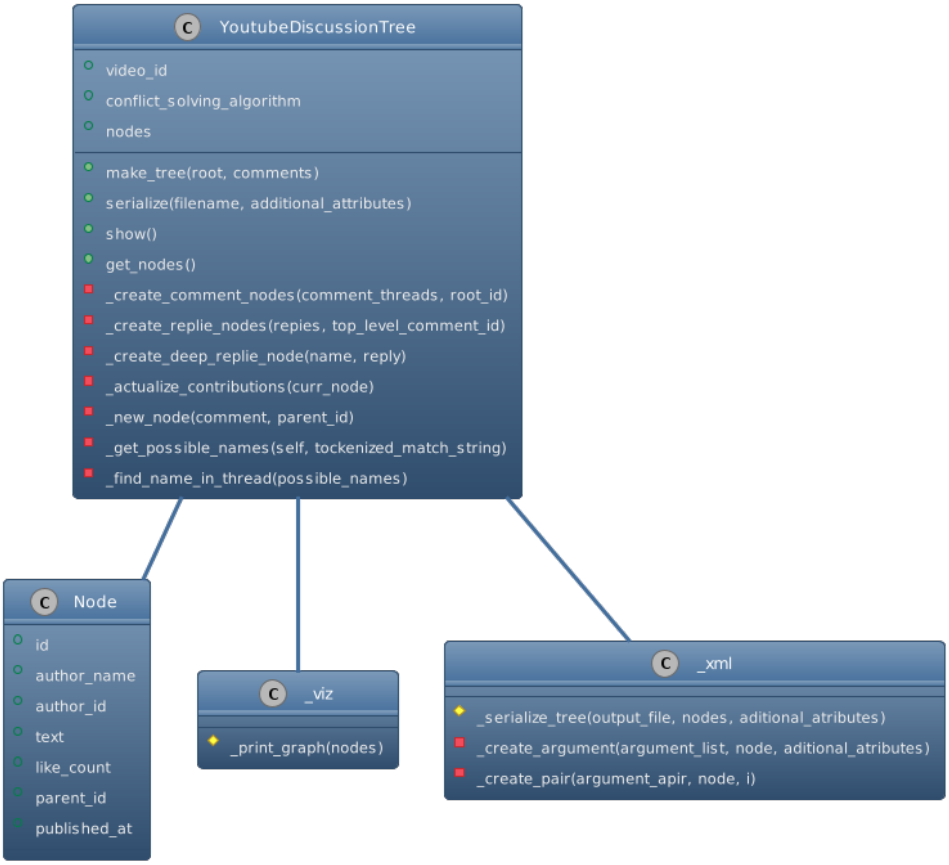

Figura 3: YoutubeDiscussionTree

La funció de generació que presenta l'objecte YoutubeDiscussionTreeAPI retorna un objecte YoutubeDiscussionTree, que com el seu nom indica, representa l'arbre de discussió que s'ha generat a partir del contingut i els comentaris d'un vídeo. Aquest és el que conté tota la lògica de generació de l'arbre, el qual està compost per un seguit d'objectes Node que estan relacionats mitjançant el paràmetre parent_id. Tots aquests nodes s'emmagatzemaran dins una llista.

Quant a l'algoritme de generació de l'arbre, primerament es construeix el node arrel a partir de la transcripció i la informació del vídeo. Seguida-

ment, s'itera sobre la llista de CommentThreads creant un Node a partir de la informació del topLevelComment. A aquest se li assigna com a pare el node arrel, i posteriorment es crea el subarbre de nodes compost per les respostes. En el processament de les respostes es fan un seguit de comprovacions. Aquelles que al principi del comentari no s'hi troba un tag "@Nom d'usuari", se'ls les hi assigna com a pare el node que representa el topLevelComment. En cas de trobar-s'hi, es mira si l'usuari al qual es respon ha realitzat més d'una contribució en la discussió que té lloc en aquest fil. En cas d'haver-ne realitzat només una, s'assigna com a node pare aquesta. Si prèviament l'usuari al qual contestem ha realitzat més d'una contribució, s'ha de fer ús de l'algoritme de resolució automàtic per tal d'inferir a quina de les contribucions d'aquest està contestant la resposta que estem processant.
Com un vídeo pot tindre molts fils de comentaris, s'ha decidit agafar els 100 més rellevants, així es limita la mida de l'arbre i s'assegura agafar aquells en els que s'hi pot generar més discussió.

A part, YoutubeDiscussionTree compta amb dues funcions per tal de representar i serialitzar aquest arbre. Aquestes operacions, al no formar part de la lògica de creació de l'arbre, s'han decidit implementar en mòduls separats.

_viz

La representació de l'arbre s'implementa mitjançant la llibreria treelib [1]. S'imprimeix per terminal l'identificador de cada un dels nodes i s'esglaona segons el nivell que ocupa a l'arbre:

```
LnX3B9oaKzw
|-- Ugg06_f0qAVH6HgCoAEC
|-- Ugg662Arr0neQXgCoAEC
|   |-- Ugg662Arr0neQXgCoAEC.8LvCxcl0tY18LvI0OyzxUW
|   |-- Ugg662Arr0neQXgCoAEC.8LvCxcl0tY18LvIRl9v4kU
```

```
|   |-- Ugg662ArrOneQXgCoAEC.8LvCxcl0tY18LvIlAkRi3e
|                     .
|                     .
|                     .
```

_xml

La serialització s'implementa utilitzant la llibreria estàndard de Python ElementTree XML [5]. L'estructura del document XML que es generarà és la següent:

```
<entailment-corpus num_edges="162" num_nodes="163">
    <argument-list>
        ...
    <argument-list>
    <argument-pairs>
        ...
    <argument-pairs>
</entailment-corpus>
```

On dins d'argument-list si llistaran tots els nodes, els quals es representaran de la següent forma:

```
<arg author="name" author_id="id autor" id="id comentari"
                                              likeCount="nº likes">
    Text comentari
</arg>
```

I dins d'argument pairs es guardaran les arestes de l'arbre de la següent manera:

```
<pair id="1">
    <t id="Ugh8N1Ch9gCr-HgCoAEC"/>
    <h id="LnX3B9oaKzw"/>
</pair>
```

On h és el destí i t l'origen.

Per la funció de serialització, també s'ha aplicat una tècnica similar a la que s'ha aplicat amb l'algoritme de resolució de conflictes a la funció de generació de l'arbre. En aquest cas, pot interessar afegir algun atribut més

a part dels que ja ofereix per defecte la llibreria als components que representen els nodes. Per tal de fer-ho, la funció serialize també actua com una funció d'alt nivell, podent passar-li per paràmetre una funció que rep un node i dona de sortida un diccionari clau valor. Aquesta funció s'executarà en el moment de serialitzar cada un dels nodes i s'afegiran els camps definits al diccionari com a atributs del component XML.

3.2.3 Interaccions amb l'API i Gestió de la Quota

_quota

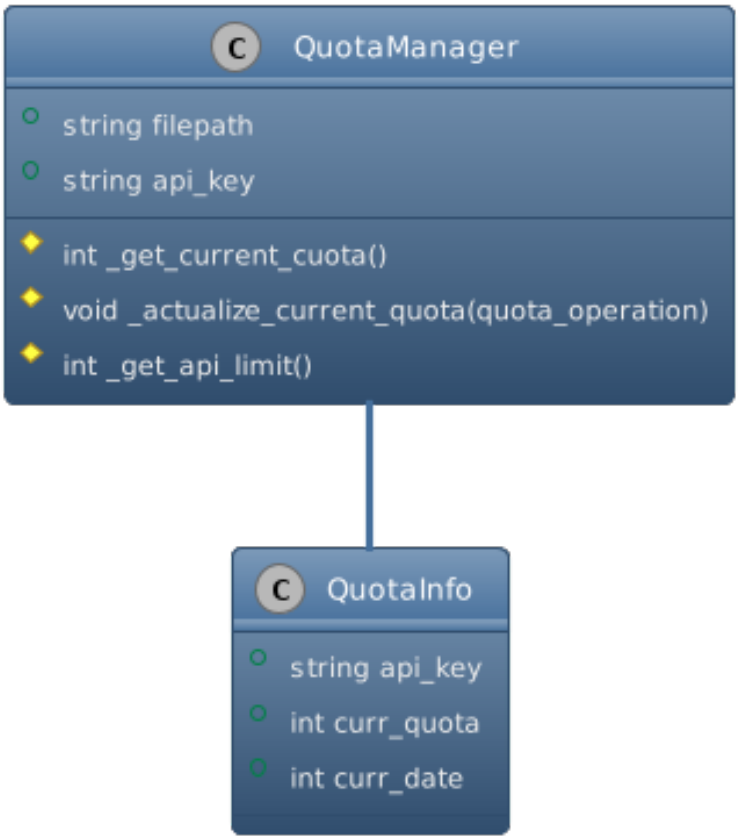

Figura 4: QuotaManager

Com s'ha comentat prèviament, YouTube no dóna la informació de

quanta quota s'ha consumit en un dia. Per tal de dur-lo, aquesta llibreria conté una implementació que comptabilitza la quota diària que es consumeix. Des de l'objecte YoutubeDiscussionTreeAPI es crea un objecte QuotaManager, el qual ens servirà per actualitzar i consultar la informació de quanta quota s'ha consumit. Per tal de persistir les dades, mitjançant la llibreria pickle [6] es serialitzarà un objecte QuotaInfo, de forma que cada cop que el vulguem actualitzar es deserialitzarà i es tornarà a serialitzar. Similarment, quan vulguem consultar qualsevol informació sobre la quota, s'haurà de deserialitzar l'objecte. Aquest s'actualitzarà cada cop que es faci una petició a la YouTube Data API mitjançant algun dels mètodes definits al mòdul _http.

_http

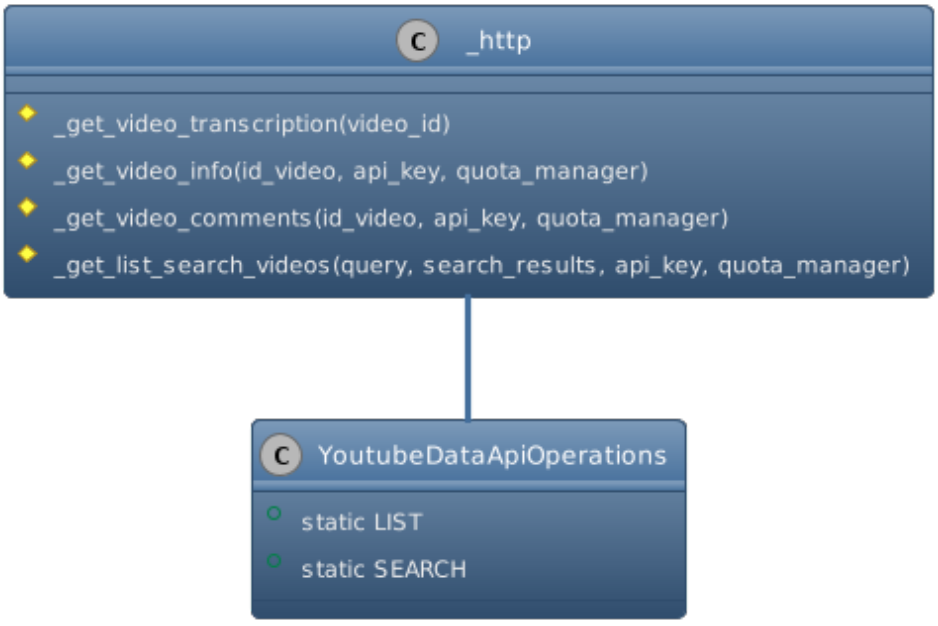

Figura 5: _http module

S'han decidit centralitzar totes les interaccions que es realitzen amb API's via http en un mòdul, _http. Aquest conté totes les funcions que

interactuen amb els diferents endpoints de la YouTube Data API, així com la funció que mitjançant la llibreria YouTube Transcript API aconsegueix les transcripcions dels vídeos.

Tal com s'ha comentat en l'anterior apartat, totes aquelles funcions que realitzen peticions a la YouTube Data API, actualitzaran l'estat de la quota cridant a la funció _actualize_current_quota d'un objecte QuotaManager que se'ls hi passarà per paràmetre des de YoutubeDiscussionTreeAPI. Les operacions disponibles i els seus costos estan definides de forma estàtica a un objecte YoutubeDataApiOperations.

3.2.4 Algoritme de Resolució de Conflictes

TF-IDF es refereix a "Term Frequency - Inverse Document Frequency" [9]. És una tècnica per a quantificar paraules que es troben en documents, generalment es computa un pes per cada una de les d'elles que representa la seva importància en el text d'un document.

Això és realitza perque, per exemple, per els humans és molt senzill entendre la frase "Aquest edifici és molt alt.", ja que coneixem les semàntiques de les paraules i la frase. Però com pot entendre l'ordinador aquesta frase? L'ordinador pot entendre qualsevol dada només quan aquesta pren la forma d'un valor numèric. Per aquesta raó, es vectoritzen els textos de forma que l'ordinador els pugui entendre millor.

Un cop vectoritzats els documents, es poden realitzar múltiples tasques així com trobar els documents més rellevants, fer un rànquing, classificacions... Aquest tipus de tècniques, són les que empra Google per al seu motor de cerca. Les pàgines són anomenades documents i el text de cerca s'anomena query. Quan realitzes una cerca mitjançant una query, Google

troba els documents més rellevants a partir d'aquesta, els ordena en termes de la rellevància i en presenta els k primers. Tot aquest procés de rànquing es realitza amb la forma vectoritzada dels documents així com també de la query. Notar però, que l'algoritme que utilitza Google és molt més sofisticat.

S'ha pensat que aquesta tècnica es pot aplicar en el cas d'ús que ocupa aquest treball. Els comentaris que són candidats a ser el node pare serien els documents. I la resposta que contesta a algun dels candidats és la query. D'aquesta forma, és realitza un rànquing dels candidats en funció de la resposta i s'obté com a node pare el més rellevant.

Quan es respon en un debat a alguna persona, tendim a utilitzar expressions i paraules similars, així és que aquest algoritme s'ha trobat adient per aquest cas d'ús.

TF-IDF

$$TF{-}IDF \ = \ TermFrequency\,(TF) * Inverse\ Document\ Frequency\,(IDF)$$

La Term Frequency és la mesura de la freqüència d'aparició d'una paraula en un document. Aquesta, depèn en la longitud del document i el general que és la paraula. Per exemple, una paraula comú com "a" pot aparèixer molts cops en un document. Però si s'agafen dos documents on un té 100 paraules, i l'altre 10.000, hi ha una probabilitat més gran que la paraula "a" aparegui més cops en el document de 10000 paraules. Però no es pot afirmar que un document per ser més llarg que un altre, sigui més important. Per tant, es realitza una normalització d'aquest valor, de forma que es divideix el nombre d'aparicions de la paraula en un document pel total de paraules d'aquest.
En aquest cas, però, l'objectiu és vectoritzar tots els documents. Si es real-

itzes aquest càlcul per cada un dels documents al tindre longituds diferents, derivarien en vectors amb longituds diferents també, i per tant no seria possible computar la seva similitud. Així doncs la normalització s'aplicarà sobre el vocabulari conjunt de tots els documents. D'aquesta forma s'assegura que els vectors que representen cada document tinguin la mateixa longitud.
El valor TF és individual per cada document i es calcularà de la següent forma:

$$tf(t,d) \; = \; recompte\ de\ t\ en\ d\ /\ long\ vocab\ total$$

on t representa els termes i d els documents.
El motiu pel qual no s'utilitzen directament aquests vectors per determinar la similitud entre documents és perquè paraules molt freqüents com "a", "de", "la" poden arribar a tindre valors molt grans i per tant l'algoritme determinaria que serien paraules molt rellevants a l'hora de determinar la similitud entre documents. Per tant, a aquest càlcul se li ha d'afegir alguna forma de relaxar els valors d'aquestes paraules que es repeteixen tant en tots els documents. Això és realitza multiplicant el valor de TF per IDF (Inverse Document Frequency).

Abans d'explicar que és l'Inverse Document Frequency, s'explicarà que és el Document Frequency. Aquest valor representa el nombre de documents en què una paraula t hi apareix. No es necessita saber el nombre de cops que el terme t apareix a cada document, només si hi apareix almenys un cop.

$$df(t) \; = \; aparicions\ de\ t\ en\ els\ documents\ /\ nombre\ total\ de\ documents$$

De la mateixa forma que amb la TF, per tal de mantenir-ho dins del rang de [0,1] es divideix el valor pel nombre total de documents.

El valor IDF és l'invers del valor DF, és a dir:

$$IDF \ = \ N/df$$

Si ens hi fixem, per paraules com "a" i "d" el seu valor serà molt baix perquè apareixeran la majoria de documents. A aquesta fórmula encara se li apliquen algunes transformacions perquè si tenim un nombre molt gran de documents, el valor de IDF es pot disparar. Per tant, per tal de mantenir-lo també en el rang de [0,1] es realitza el logaritme.

$$IDF \ = \ log(N/(df + 1))$$

El (+1) en el denominador es fica per assegurar que mai dividirem entre 0. D'aquesta forma, queda que:

$$tf - idf(t, d) \ = \ tf(t, d) \ * \ log(N/(df + 1))$$

Ara sí, mitjançant aquests vectors, es pot calcular la similitud d'aquests amb la query d'entrada. Per la query realitzarem el mateix procés, es calcularà el vector de valors tf-idf mitjançant el mateix vocabulari per tal que la llargada sigui la mateixa, i aquest cop sumant 1 també al nombre de documents, ja que la query es considera també un document. Un cop obtingut es realitzarà el càlcul de la "Cosine Similarity". Aquest càlcul representa tots els vectors dels documents en un gràfic on tots parteixen des del centre, per exemple una representació possible seria la que es troba a la figura 6.

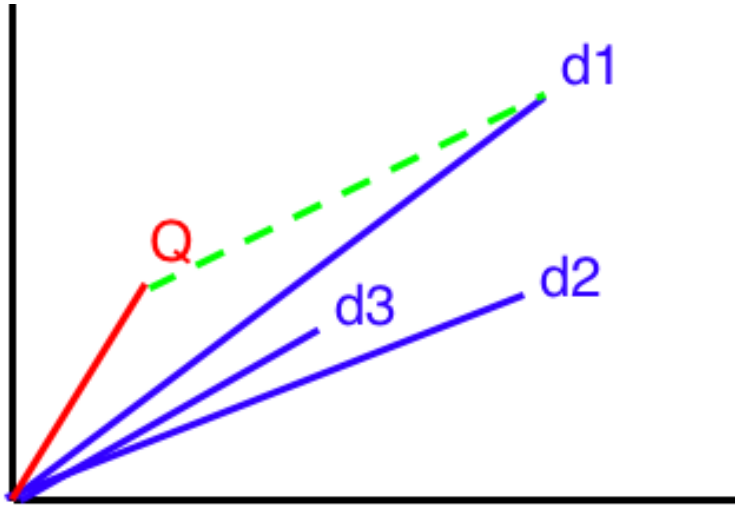

Figura 6: Cosine Similarity

Un cop representats retorna aquell candidat el qual l'angle és menor respecte amb el vector de la query. En el cas de la figura serà d1.

4 Ús del sistema

4.1 Instal · lació

Es recomana instal · lar-ho a través de pip:

```
pip install youtube_discussion_tree_api
```

Si es vol utilitzar des del codi font s'hauran d'instal · lar els requeriments:

```
pip install -r requeriments.txt
```

4.2 API

En aquest apartat s'explicaran les funcionalitats que presenta l'API i com se'n fa ús.

4.2.1 Generació de l'arbre

Aquesta és la funcionalitat principal. La forma més senzilla per generar un arbre és:

```
from youtube_discussion_tree_api
                import YoutubeDiscussionTreeAPI


api = YoutubeDiscussionTreeAPI("<GCP project api key>")
tree = api.generateTree("videoId")
```

Primerament es crea un objecte YoutubeDiscussionTreeAPI amb l'api key del teu projecte de GCP. Seguidament es crida al mètode generateTree, el qual donarà com a resposta un objecte YoutubeDiscussionTree que representarà l'arbre de discussió.
En cas que es volgués resumir la transcripció del contingut del vídeo, es podria cridar a la funció generateTree de la següent forma:

```
tree = api.generateTree("videoId", summarization=True)
```

De la mateixa manera, si es volgués passar per paràmetre un algoritme de resolució de conflictes:

```
tree = api.generateTree("videoId",
        conflict_solving_algorithm=nom_func_algoritme)
```

4.2.2 Serialització de l'arbre

La forma més senzilla de serialitzar un arbre és:

```
tree.serialize("outuput_file.xml")
```

Per altra banda, se li pot passar per paràmetre una funció per afegir camps addicionals als components que representen els nodes:

```
def my_additional_atributes(node):
    return {
        "date" : node.publishedAt
        "sentiment" : sentiment_analysis(node.txt)
```

```
    }


tree.serialize("output_file.xml",
                           my_aditional_atributes)
```

Si es passa per paràmetre aquesta funció, els nodes en el fitxer XML presentaran la següent estructura:

```
<arg author="name" author_id="id_author" id="id_comment"
          likeCount="nº likes" date="dd-mm-yyyy" sentiment="NEGATIVE">
    Text comentari.
</arg>
```

4.2.3 Representació de l'arbre

Per tal de representar l'arbre s'ha d'escriure:

```
tree.show()
```

4.2.4 Llista de Nodes

Per tal d'aconseguir la llista de nodes que conformen l'arbre escriure:

```
tree.get_nodes()
```

D'aquesta forma és retornara una llista amb objectes Node:

```
class Node:
    id: int
    author_name: str
    author_id: int
    text: str
    like_count: int
    parent_id: int
    published_at : str
```

4.2.5 Búsqueda de vídeos

A través de l'objecte YoutubeDiscussionTreeAPI es pot realitzar una cerca de vídeos de la següent forma:

```
from youtube_discussion_tree_api
                        import YoutubeDiscussionTreeAPI


api = YoutubeDiscussionTreeAPI("<api_key>")
videos = api.search_videos("Functional Programming")
```

D'aquesta forma es retornarà una llista amb objectes Vídeo:

```
class Video:
    id: int
    title : str
    description: str
    channel_name: str
    channel_id: str
    published_at : str
```

Si es volgués realitzar una cerca amb més resultats:

```
from youtube_discussion_tree_api
                        import YoutubeDiscussionTreeAPI


api = YoutubeDiscussionTreeAPI("<api_key>")
videos = api.search_videos("Functional Programming",50)
```

Es pot determinar el nombre de resultats que es volen obtenir en un rang entre 1 i 50.

4.2.6 Consultar Quota

Per tal de consultar la quota diària que s'ha consumit utilitzant la llibreria:

```
from youtube_discussion_tree_api
                        import YoutubeDiscussionTreeAPI


api = YoutubeDiscussionTreeAPI("<api_key>")
api.quota_info()
```

Això retornarà un diccionari:

```
{
```

```
    "limit" : 10000
    "spent" : <num-expended-quota>
}
```

5 Avaluació

Abans d'avaluar i comentar els resultats que obté l'algoritme de construcció d'arbres de discussió que s'ha implementat, es mostrarà un exemple representatiu de com funciona aquest amb un fragment d'una discussió.

Primerament l'algoritme realitza una transcripció del vídeo i mitjançant l'API, n'obté la informació restant per tal de generar el node arrel. Posteriorment, comença el processament dels comment threads. Imaginem que s'està processant el commentThread que té com a comentari principal el representat en la figura 7.

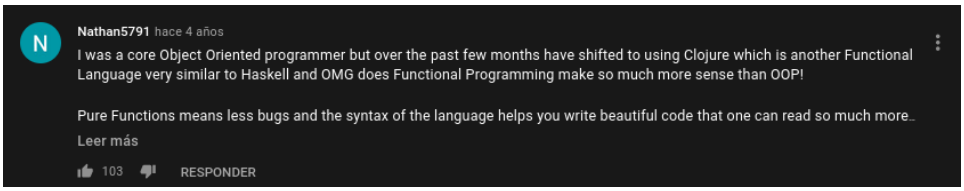

Figura 7: Comentari Principal del Fil

El pare d'aquest comentari serà el node arrel, i els fills tots aquells que el contesten. Seguidament, començarà el processament del subarbre que representarà al fil de comentaris. En cas de trobar-se amb comentaris com els representats en la figura 8, s'assignaria com a pare directament el node principal del fil, ja que no s'etiqueta a cap usuari i per tant no estan contestant a ningú en concret de les respostes sinó al comentari principal.

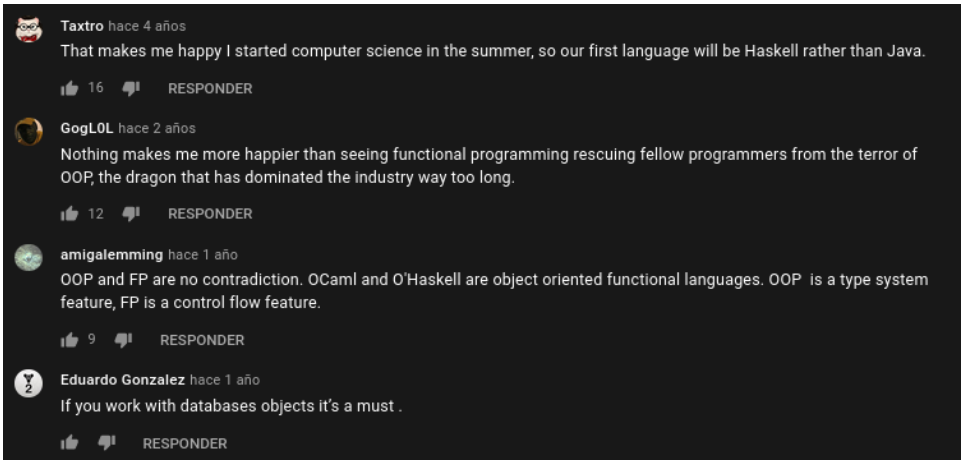

Figura 8: Resposta Normal

Un cop es troba un comentari en el qual s'etiqueta a un altre usuari com es representa en la figura 9, es mira quantes contribucions ha fet prèviament aquest al fil.

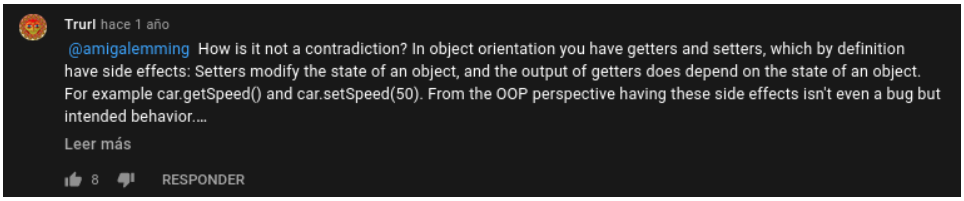

Figura 9: Resposta amb etiquetatge

En aquest cas l'usuari al qual s'etiqueta només ha fet una contribució prèvia a aquest comentari i per tant podem inferir que es contesta a aquesta. En cas de trobar-se, però, que l'usuari ha realitzat més d'una contribució en el fil tal com representa la figura 10, s'utilitza l'algoritme de resolució de conflictes basat en el TF-IDF per ranquejar i triar al candidat al qual es creu que el comentari contesta.

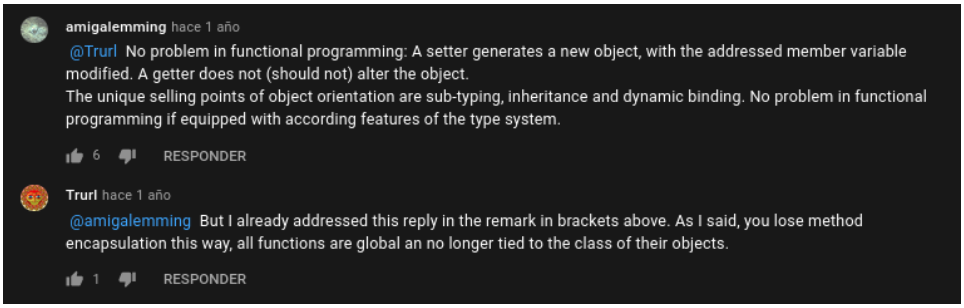

Figura 10: Resposta amb etiquetatge i conflicte

Per tal de fer una avaluació de l'algoritme de resolució automàtic, s'ha decidit implementar un script que utilitza la llibreria. En aquest algoritme es realitzarà una cerca de vídeos sobre un camp determinat i posteriorment es construiran dos arbres de forma paral · lela per cada un dels vídeos. Un dels arbres resoldrà els conflictes que es trobi de forma automàtica amb l'algoritme que conté la llibreria, i per l'altre s'ha dissenyat un algoritme interactiu per tal de resoldre els conflictes de forma manual. Aquest últim printarà els conflictes per terminal i serà l'usuari de l'script l'encarregat de resoldre'ls. Finalment es compararà com s'han resolt els conflictes en ambdós casos per veure si s'han resolt de la mateixa manera.

Aquest mecanisme de comprovació supervisat assumeix que l'usuari de l'script té la veritat absoluta, és a dir, la forma en com resolgui els conflictes es considerarà la veritat. S'ha decidit fer-ho així pel fet que YouTube no ofereix la possibilitat de saber la veritat, així doncs, el més proper que es pot estar d'aquesta és la interpretació que realitzem els humans sobre la conversa.

Després de vàries execucions, s'ha observat que el percentatge d'encert no és massa elevat. Com s'han anat resolent els conflictes de forma manual s'han pogut observar diversos factors que poden ser els causants que aquest algoritme no sigui gaire precís. El principal és que en molts casos, les respostes són molt curtes i pràcticament no hi han text per a generar els vectors. De

fet, s'ha observat que en les respostes on hi ha més discussió i raonament, l'algoritme funciona molt millor.

Aquestes proves com bé s'ha dit s'han realitzat a l'engròs mitjançant el mètode de cerca de la implementació. Però, per altra banda, també s'ha pogut observar que és més efectiu sobre vídeos en els què aparentment en els comentaris hi ha més discussió de qualitat. Així que, si s'és més selectiu amb els vídeos dels quals es vol generar l'arbre de discussió, aquest presenta bons resutats.

6 Conclusions

Tal com es va anunciar als objectius, s'ha implementat una solució la qual permet generar un arbre de discussió a partir dels comentaris que es troben en un vídeo de YouTube. Aquesta, és una eina general i a la vegada compleix els requisits per poder ser utilitzada per l'analitzador de converses desenvolupat pel grup de recerca GREiA de la Universitat de Lleida.

Alguns dels objectius, també, han estat que aquesta implementació fos usable i de fàcil accés, així que s'ha implementat una llibreria la qual es troba pública al repositori de software de tercers de Python [8]. Aquesta, a més, compta amb un disseny centrat en l'usuari per tal de fer-ne senzill el seu ús.

Quant a l'algoritme de resolució de conflictes, tal com s'ha comentat, en termes generals no té un percentatge d'encert massa elevat. Així i tot, si s'utilitza d'una forma més selectiva i es tria generar arbres a partir de vídeos amb discussió de qualitat als comentaris, el funcionament d'aquest millora molt.

La realització d'aquest treball ha estat una experiència molt enriquidora. Primerament per la fase d'anàlisi exhaustiu que es va realitzar sobre el domini del problema. Seguidament, per l'adaptació que es va realitzar quan es va trobar el conflicte de simplificació de dades que presentava la YouTube Data API. I finalment per la part de disseny i implementació d'una solució que facilita la generació d'arbres de discussió i que resol aquesta problemàtica trobada.

Es creu que aquesta llibreria, a més d'una eina d'ajut per a la recerca i per a l'anàlisi d'una de les xarxes socials més grans d'avui dia, també proposa una solució molt interessant a un problema que pot afectar a futurs

usuaris de la YouTube Data API.

De totes formes, aquesta llibreria encara pot millorar. De fet, en el futur es planeja seguir mantenint-la i aplicant millores a l'algoritme de resolució de conflictes. També es planeja afegir més mòduls i modificar la implementació per tal de fer-la encara més flexible i oberta de cara a l'usuari final.

7 Bibliografia

[1] Xiaming Chen. *Treelib*. URL: https://treelib.readthedocs.io/en/latest/.

[2] Travis CI. *Travis CI*. URL: https://travis-ci.com/.

[3] Coveralls. *Coveralls*. URL: https://coveralls.io/.

[4] Jonas Depoix. *Youtube Transcript API*. URL: https://github.com/jdepoix/youtube-transcript-api/.

[5] Python Standard Library. *Element Tree XML*. URL: https://docs.python.org/3/library/xml.etree.elementtree.html.

[6] Python Standard Library. *Pickle*. URL: https://docs.python.org/3/library/pickle.html.

[7] The HuggingFace Project. *Transformers*. URL: https://huggingface.co/transformers/index.html.

[8] Quim10^{-}12. *Youtube Discussion Tree API*. URL: https://pypi.org/project/youtube-discussion-tree-api/.

[9] William Scot. *TF-IDF from scratch in python on real world dataset*. URL: https://towardsdatascience.com/tf-idf-for-document-ranking-from-scratch-in-python-on-real-world-dataset-796d339a4089.

[10] YouTube. *Youtube Data API*. URL: https://developers.google.com/youtube/v3.

[11] YouTube. *Youtube Data API CommentThreads*. URL: https://developers.google.com/youtube/v3/docs/commentThreads/list.

[12] YouTube. *Youtube Data API Search*. URL: https://developers.google.com/youtube/v3/docs/search.

[13] YouTube. *Youtube Data API Videos*. URL: https://developers.google.com/youtube/v3/docs/videos.